

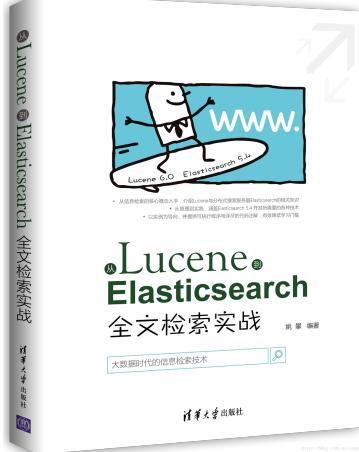
姚攀的博客 1.01^365=31.78

Engineers are versatile minds who create links between science, technology, and society.

[目录视图](#)
[摘要视图](#)
[RSS 订阅](#)

《从Lucene到Elasticsearch：全文检索实战》

希望本书能对您的工作和学习带来帮助，感谢支持！



当当 京东 亚马逊 天猫

Lucene、ES、ELK开发交流群:370734940

[加入QQ群](#)

个人资料



yyyseb

关注

发私信



访问: 844276次

积分: 8642

等级: BLOC 6

排名: 第2544名

图灵赠书——程序员11月书单 【思考】Python这么厉害的原因竟然是！ 感恩节赠书：《深度学习》等异步社区优秀图书和作译者评选启动！ 每周荐书：京东架构、Linux内核、Python全栈

Spring全家桶(一)HelloWorld与入门基础

标签: [spring](#) [helloworld](#)

2017-05-16 11:23

2026人阅读

评论(0)

分类:

[spring \(8\)](#)

版权声明: 本文为博主原创文章,未经博主允许禁止转载(<http://blog.csdn.net/napoay>)

[目录\(?\)](#)

[+]

一、认识Spring

1.1 Spring简介

Spring是一个开源框架，为简化企业级应用而生，是一个IOC(DI)和AOP容器框架。IOC和AOP不太好理解，这里有一篇文章值得参考 [IOC和AOP的一些基本概念](#)

IOC(Inversion of Control):

反转资源获取的方向，传统的资源查找方式要求组件向容器发起请求查找资源，容器适时返回资源作为回应。应用IOC之后，容器主动地将资源推送给它所管理的组件，组件要做的就是选择一种合适的方式接受资源，这种行为也被称为查找的被动形式。

DI(Dependency Injection):

IOC的另一种表述方式，即组件以一些预先定义好的方式（比如setter方法）接受来自容器的资源注入。相对于IOC而言，DI在表述上更直接。

原创: 202篇

转载: 2篇

译文: 6篇

评论: 451条

我的课程

Elasticsearch 5.4新闻搜索项目实战

StackOverFlow

<http://stackoverflow.com/users/6526424>

统计

阅读排行

[搜索]ElasticSearch Java Api(...	(61756)
Elasticsearch索引mapping的写...	(36806)
Elasticsearch 5.X下JAVA API使...	(31047)
搭建Elasticsearch 5.4分布式集群	(29631)
Elasticsearch 5.1.1 head插件安...	(25822)
Elasticsearch java api(五) Bulk...	(21822)
ElasticSearch Java Api(四) -删...	(19887)
mac命令行启动tomcat	(18615)
Elasticsearch 5 Ik+pinYin分词...	(17900)
Elasticsearch shield权限管理详解	(17369)

博客专栏



Spring全家桶

文章: 10篇

阅读: 10615



MapReduce编程



文章: 7篇

阅读: 24315



Lucene&Elasticsearch&ELK专栏

文章: 42篇

阅读: 461194



数据库

文章: 9篇

阅读: 18146



J2EE

文章: 24篇

阅读: 92769



ruby on rails

文章: 18篇

阅读: 37367

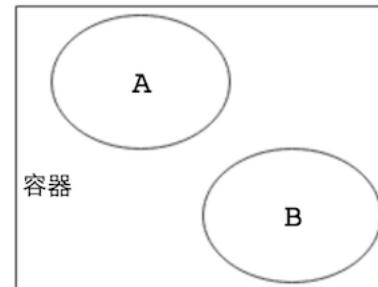
数据结构与算法

```

class A{
}

class B{
    private A a;
    public void setA(A a){
        this.a=a;
    }
}

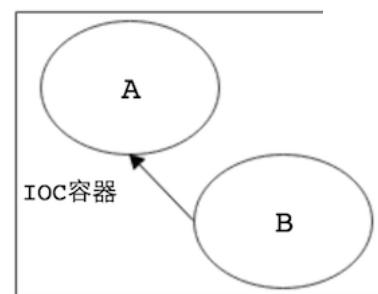
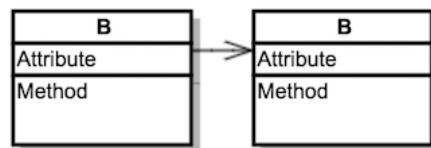
```



```

A a=getA()
B b=getB()
b.setA(a)

```



B b=getB()
<http://blog.csdn.net/napoay>

1.2 Spring特点

1. 轻量级: Spring是非侵入的，基于Spring开发的应用中的对象可以不依赖Spring的API
2. 依赖注入
3. 面向切面编程
4. 容器: Spring是一个容器，它包含并且管理应用对象的生命周期
5. 框架: Spring使用简单的组件配置组合成一个复杂的应用，在Spring中可以使用XML和Java注解组合这些对象
6. 一站式: Spring是一个一站式的框架，在IOC和AOP的基础上可以整合企业应用的开源框架和第三方类库。使用Spring可以把Java EE中的知识点都包括起来，Spring等同于Java EE。

1.3 Spring模块

Spring框架包含了大约20个模块，如下图所示。这些模块再细分成Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, Messaging, and Test。



文章: 13篇
阅读: 17565

文章分类

Elasticsearch (43)
Lucene (15)
hadoop (16)
J2EE (35)
数据结构 (16)
前端 (16)
ruby (18)
机器学习 (4)
python (5)
linux (11)
latex (2)
数据库 (12)
spring (9)
tools (1)
hibernate (1)
mybatis (1)
生活 (1)

文章存档

2017年12月 (9)
2017年11月 (1)
2017年10月 (3)
2017年09月 (2)
2017年08月 (1)

展开

最新评论

MapReduce编程(六) 从HDFS导入数据到Elasticsearch
qq_38094271 :用最新版本的那个 成功了。
注意json文件最后一行要换行
《从Lucene到Elasticsearch:全文检索实战...
牙小木 :书已买，正在看
Lucene扩展停用词字典与自定义词库
yyysyb :@cuipeng6561:这个文件在ik的目录下。
Lucene扩展停用词字典与自定义词库
cuipeng6561 :IKAnalyzer.cfg.xml 这个文件是要自己写么？

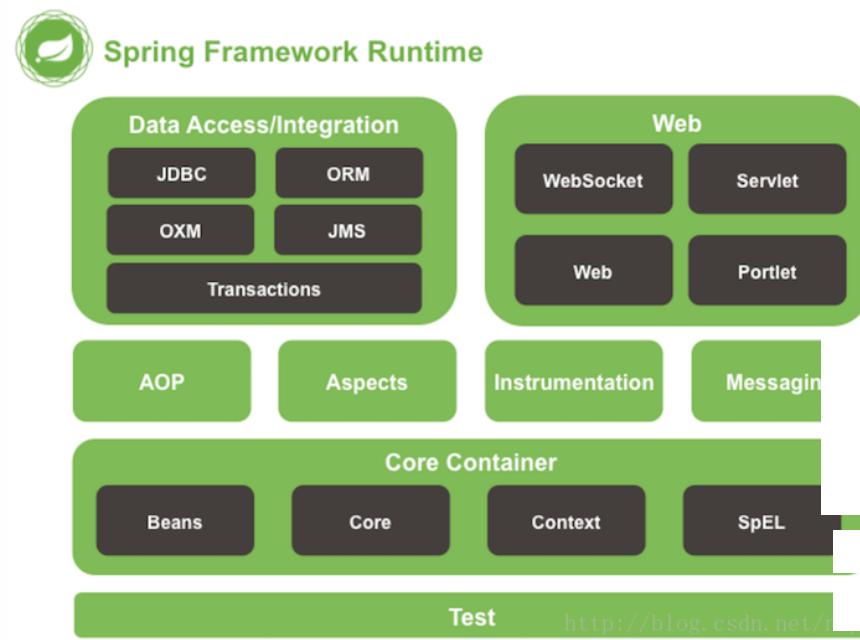
新人千万不要在 Windows 上使用 Ruby on ...
韩大嘴 :对于新手的我来说的，相当中肯！

《从Lucene到Elasticsearch:全文检索实战...
艾鹤 :6666，恭喜发财。

Elasticsearch 5.X下JAVA API使用指南
qq_41296216 :@napoay:maven坐标是 <dependency> ...

Elasticsearch java api(五) Bulk批量索引
yyysyb :@wangmaohong0717:可以不指定，会自动生成。

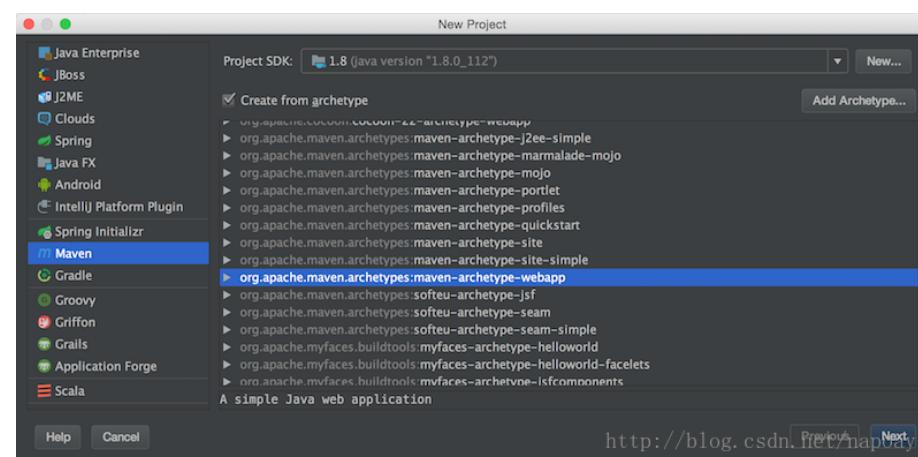
Elasticsearch 5.X下JAVA API使用指南
yyysyb :@qq_41296216:贴一下maven坐标吧，或者，博客上有QQ群，加群讨论



二、Spring HelloWorld

2.1 搭建开发环境

在IDEA中新建Maven工程，选择maven-archetype-webapp。



在pom.xml中添加spring的maven依赖：

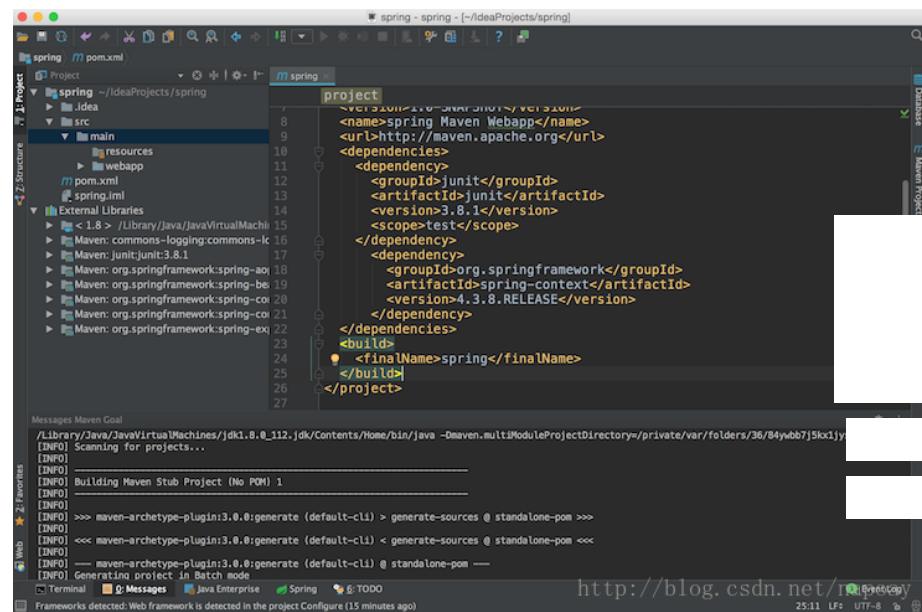
```

1 <dependency>
2   <groupId>org.springframework</groupId>
3   <artifactId>spring-context</artifactId>
4   <version>4.3.8.RELEASE</version>
5 </dependency>

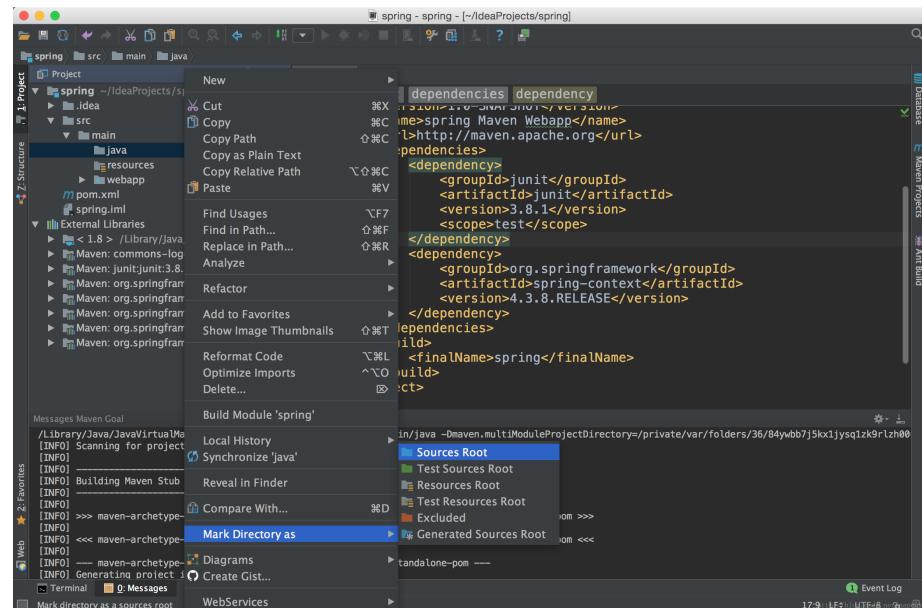
```

Elasticsearch java api(五) Bulk批量索引
hello-friend : 批量索引数据的时候, 必须指定_id吗

导入完成以后, 在External Libraries目录下可以看到新导入的五个jar包:spring-aop、spring-beans、spring-contex、spring-core、spring-expression。



在src/main目录下新建Directory, 设置资源文件夹。



在java目录下新建一个包, 再增加一个HelloWorld类。HelloWorld类中设置了一个私有域name, 一个setName方法和一个sayHello方法。

```

1 package com.stuspring;
2
3 /**
4 * Created by bee on 17/4/22.
5 */
6 public class HelloWorld {
7     private String name;

```

```

8
9     public void setName(String name) {
10         this.name = name;
11     }
12
13     public void sayHello(){
14         System.out.println("Hello:"+name);
15     }
16 }
```

在新增一个Main类做测试,创建一个HelloWorld对象, 设置name, 调用sayHello

```

1 package com.stuspring;
2
3 /**
4 * Created by bee on 17/4/22.
5 */
6 public class Main {
7     public static void main(String[] args) {
8         HelloWorld hw1=new HelloWorld();
9         hw1.setName("Doraemon");
10        hw1.sayHello();
11    }
12 }
```

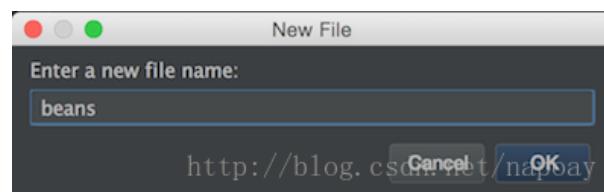
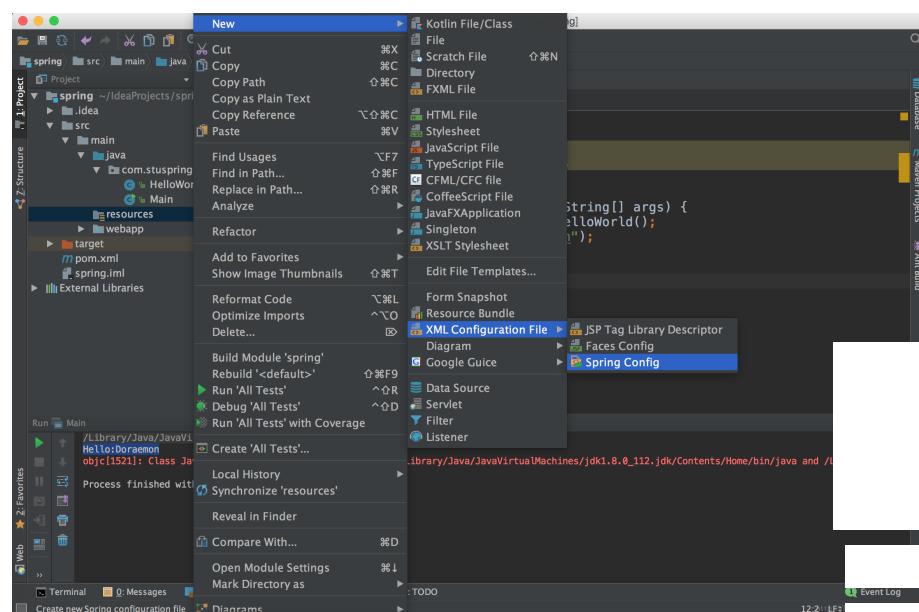
运行结果:

```
1 Hello:Doraemon
```

上面是传统的创建对象的方法, 下面通过Spring来创建HelloWorld的对象。

2.2 使用Spring管理对象

首先创建Spring的配置文件, 点击resources文件夹, 右键->New->XML Configuration File->Spring Config, 如下如所示。文件名为beans,

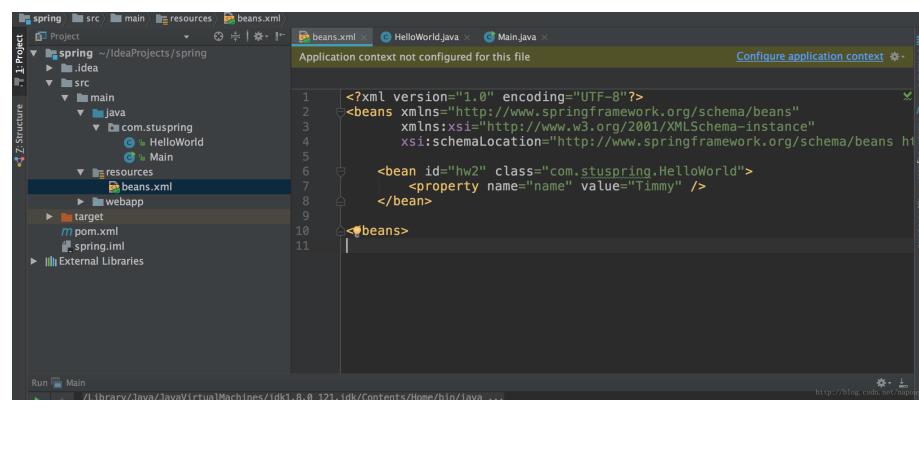


点击OK以后，会在resources目录下生成beans.xml。

在beans.xml定义一个新的HelloWorld对象，方式如下：

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4          xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.
5
6      <bean id="hw2" class="com.stuspring.HelloWorld">
7          <property name="name" value="Timmy" />
8      </bean>
9
10 </beans>
```



在Main类中获取id为hw2的HelloWorld对象：

```

1 package com.stuspring;
2
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6 /**
7 * Created by bee on 17/4/22.
8 */
9 public class Main {
10     public static void main(String[] args) {
11
12         //1.创建一个HelloWorld对象
13         HelloWorld hw1=new HelloWorld();
14         //2.对name属性赋值
15         hw1.setName("Doraemon");
16         //3.调用sayHello方法
17         hw1.sayHello();
18
19
20         //1.创建IOC容器对象
21         ApplicationContext ctx=new ClassPathXmlApplicationContext("beans.xml");
22
23         //2.从IOC容器中获取对象
24         HelloWorld hw2= (HelloWorld) ctx.getBean("hw2");
25         //3.调用sayHello方法
26         hw2.sayHello();
27     }
28 }
```

再次运行，打印结果：

```

1 Hello:Doraemon
2 Hello:Timmy
```

ClassPathXmlApplicationContext用于加载CLASSPATH下的配置文件，传入参数为文件名，这里就是加载resources目录下的beans.xml，通过ApplicationContext对象的getBean方法获取对象，传入参数为配置文件中bean的id。

在创建IOC容器对象的时候，所有Bean实例都已经初始化完成。可以通过一个简单的方法进行测试，在HelloWorld类的setName方法中打印一行提示信息，增加一个无参构造方法：

```

1 package com.stuspring;
2
3 /**
4 * Created by bee on 17/4/22.
5 */
6 public class HelloWorld {
7     private String name;
8
9     public void setName(String name) {
10         System.out.println("setName:"+name);
```

```

11         this.name = name;
12     }
13
14     public void sayHello(){
15         System.out.println("Hello:"+name);
16     }
17
18     public HelloWorld(){
19         System.out.println("HelloWorld的无参构造方法!");
20     }
21 }
```

在Main类的Main函数中只创建一个ApplicationContext对象：

```

1 package com.stuspring;
2
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.support.ClassPathXmlApplicatio
5
6 /**
7 * Created by bee on 17/4/22.
8 */
9 public class Main {
10     public static void main(String[] args) {
11         ApplicationContext ctx=new ClassPathXmlApplicationContext("beans.xml");
12     }
13 }
```

运行，打印结果如下：

```

1 HelloWorld的无参构造方法!
2 setName:Timmy
```

三、 Bean配置细节

Bean的配置形式：

1. 基于XML文件的方式
2. 基于注解的方式

Bean的配置方式：

1. 通过全类名(反射)
2. 通过工厂方法(静态工厂方法、实例工厂方法)
3. FactoryBean

依赖注入的方式：

1. 属性注入

2. 构造器注入

Spring容器:

1. BeanFactory: IOC容器的基本实现，是Spring框架的基础设施，面向Spring本身
2. ApplicationContext: 提供了更高级的特性，是BeanFactory的子接口，面向Spring框架的开发者，几乎所有的应用场合都直接使用ApplicationContext而非底层的BeanFactory。

3.1 构造器注入

新建一个Car类，四个属性、2个重载的构造方法、一个toString()方法。

```

1 package com.stuspring;
2
3 /**
4 * Created by bee on 17/4/23.
5 */
6 public class Car {
7
8     private String brand;
9     private String corp;
10    private double price;
11    private int maxSpeed;
12
13    public Car(String brand, String corp, int maxSpeed) {
14        this.brand = brand;
15        this.corp = corp;
16        this.maxSpeed = maxSpeed;
17    }
18
19    public Car(String brand, String corp, double price) {
20        this.brand = brand;
21        this.corp = corp;
22        this.price = price;
23    }
24
25    @Override
26    public String toString() {
27        return "Car{" +
28                "brand='" + brand + '\'' +
29                ", corp='" + corp + '\'' +
30                ", price=" + price +
31                ", maxSpeed=" + maxSpeed +
32                '}';
33    }
34}

```

配置bean:

```

1 <!--使用构造器注入属性可以指定参数的位置和参数的类型以区分重载的构造器-->
2 <bean id="car1" class="com.stuspring.Car">
3     <constructor-arg name="brand" value="Ford" index="0"/>

```

```

4      <constructor-arg name="corp" value="ShangHai" index="1"/>
5      <constructor-arg name="price" value="100000" index="2" />
6    
```

7

```

8      <!--属性值可以通过value子节点进行配置，特殊字符可以使用<![CDATA[ ]]>包裹-->
9      <bean id="car2" class="com.stuspring.Car">
10     <constructor-arg name="brand" value="Audi"/>
11     <constructor-arg name="corp" >
12       <value><![CDATA[<Shanghai>]]</value>
13     </constructor-arg>
14     <constructor-arg name="maxSpeed" value="280"/>
15   
```

16

```

17   <bean id="car3" class="com.stuspring.Car">
18     <constructor-arg value="DaZhong"/>
19     <constructor-arg value="ShangHai"/>
20     <constructor-arg type="int">
21       <value>300</value>
22     </constructor-arg>
23   
```

3.2 Bean之间的引用

应用程序之间的Bean经常需要相互协作以完成应用程序的功能，要使Bean之间能相互访问，就必须在Bean配置文件中指定对Bean的引用。可以通过ref属性指定Bean的引用。

新建一个Person类，有姓名、年龄、拥有的汽车三个属性：

```

1 package com.stuspring;
2
3 /**
4  * Created by bee on 17/4/24.
5 */
6 public class Person {
7     private String name;
8     private int age;
9     private Car car;
10    //省略setter、getter和toString()方法
11
12    public Person() {
13    }
14
15    public Person(String name, int age, Car car) {
16        this.name = name;
17        this.age = age;
18        this.car = car;
19    }
20 }

```

在beans.xml中创建Person对象：

```

1 <bean id="person1" class="com.stuspring.Person">
2   <property name="name" value="LiHua"/>

```

```

3      <property name="age" value="28"/>
4      <property name="car" ref="car1"/>
5      </bean>
6      <bean id="person2" class="com.stuspring.Person">
7          <property name="name" value="ZhaoHua"/>
8          <property name="age" value="29"/>
9          <property name="car">
10         <ref bean="car2"/>
11     </property>
12 </bean>
13
14 <!--内部bean,不能被外部引用, 只能在内部使用-->
15 <bean id="person3" class="com.stuspring.Person">
16     <property name="name" value="LiSi"/>
17     <property name="age" value="31"/>
18     <property name="car">
19         <bean class="com.stuspring.Car">
20             <constructor-arg name="brand" value="Ford"/>
21             <constructor-arg name="corp" value="ShangHai"/>
22             <constructor-arg name="price" value="20000"/>
23         </bean>
24     </property>
25 </bean>
26 <bean id="person4" class="com.stuspring.Person">
27     <constructor-arg name="name" value="Jerry"/>
28     <constructor-arg name="age" value="25"/>
29     <constructor-arg name="car" ref="car3"/>
30 </bean>
31
32 <!--测试null值-->
33 <bean id="person5" class="com.stuspring.Person">
34     <constructor-arg name="name" value="John"/>
35     <constructor-arg name="age" value="31"/>
36     <constructor-arg name="car"><null/></constructor-arg>
37 </bean>
38
39 <!--级联属性赋值,需要对相应的属性提供setter方法。注意: 属性需要先初始化以后再为级联属性赋值-->
40 <bean id="person6" class="com.stuspring.Person">
41     <constructor-arg name="name" value="Linda"/>
42     <constructor-arg name="age" value="33"/>
43     <constructor-arg name="car" ref="car1"/>
44     <property name="car.maxSpeed" value="240"/>
45 </bean>

```

3.2 Bean的集合属性

集合属性赋值:在xml中可以通过内置的xml标签, 例如<List>、<set>、<map>来配置集合属性。

3.2.1 Set属性赋值

一个Person有多个汽车, 改进Person类:

```
1 package com.stuspring.collections;
2 import com.stuspring.Car;
3
4 import java.util.List;
5
6 /**
7 * Created by bee on 17/4/24.
8 */
9 public class Person {
10     private String name;
11     private int age;
12     private List<Car> cars;
13
14     public String getName() {
15         return name;
16     }
17
18     public int getAge() {
19         return age;
20     }
21
22     public List<Car> getCars() {
23         return cars;
24     }
25
26     public void setName(String name) {
27         this.name = name;
28     }
29
30     public void setAge(int age) {
31         this.age = age;
32     }
33
34     public void setCars(List<Car> cars) {
35         this.cars = cars;
36     }
37
38     public Person() {
39     }
40
41     public Person(String name, int age, List<Car> cars) {
42         this.name = name;
43         this.age = age;
44         this.cars = cars;
45     }
46
47     @Override
48     public String toString() {
49         return "Person{" +
50                 "name='" + name + '\'' +
51                 ", age=" + age +
52                 ", cars=" + cars +
53                 '}';
54     }
55
56 }
```

57

}

在beans.xml中创建bean:

```

1   <bean id="person7" class="com.stuspring.collections.Person">
2       <property name="name" value="Mike"/>
3       <property name="age" value="29"/>
4       <property name="cars">
5           <list>
6               <ref bean="car1"/>
7               <ref bean="car2"/>
8               <bean class="com.stuspring.Car">
9                   <constructor-arg value="DaZhong"/>
10                  <constructor-arg value="ShangHai"/>
11                  <constructor-arg type="int">
12                      <value>300</value>
13                  </constructor-arg>
14              </bean>
15          </list>
16      </property>
17  </bean>
```

把集合独立出来，可以共享被多个bean引用(需要导入util命名空间):

```

1  <!--导入util命名空间 -->
2  <util:list id="cars">
3      <ref bean="car1"/>
4      <ref bean="car2"/>
5  </util:list>
6
7  <bean id="person9" class="com.stuspring.collections.Person">
8      <property name="name" value="NewPerson"/>
9      <property name="age" value="29"/>
10     <property name="cars" ref="cars"></property>
11  </bean>
```

3.2.2 Map属性赋值

```

1 package com.stuspring.collections;
2
3 import com.stuspring.Car;
4
5 import java.util.Map;
6
7 /**
8 * Created by bee on 17/4/24.
9 */
10 public class NewPerson {
11     private String name;
12     private int age;
13
14     private Map<String,Car> cars;
//省略setter、getter、无参构造方法、有参构造方法和toString方法
}
```

```

15
16 }

```

配置bean:

```

1   <bean id="person8" class="com.stuspring.collections.NewPerson">
2     <property name="name" value="Tina"/>
3     <property name="age" value="26"/>
4     <property name="cars">
5       <map>
6         <entry key="AA" value-ref="car1"/>
7         <entry key="BB" value-ref="car2"/>
8       </map>
9     </property>
10   </bean>

```

3.2.3 Properties属性赋值

新建一个DataSource类模拟数据库连接:

```

1 package com.stuspring.collections;
2
3 import java.util.Properties;
4
5 /**
6  * Created by bee on 17/4/24.
7  */
8 public class DataSource {
9
10    private Properties properties;
11
12    public void setProperties(Properties properties) {
13        this.properties = properties;
14    }
15
16    public Properties getProperties() {
17        return properties;
18    }
19
20    @Override
21    public String toString() {
22        return "DataSource{" +
23            "properties=" + properties +
24            '}';
25    }
26 }

```

在beans.xml中新建bean:

```

1   <!-- 配置properties-->
2   <bean id="dataSource" class="com.stuspring.collections.DataSource">
3     <property name="properties">
4       <props>

```

```

5      <prop key="user">root</prop>
6      <prop key="password">123456</prop>
7      <prop key="jdbcUrl">jdbc:mysql:///test</prop>
8      <prop key="driverClass">com.mysql.jdbc.Driver</prop>
9      </props>
10     </property>
11   </bean>

```

四、命名空间

Spring整合了各种工具，为了简化操作，spring提供了对各种工具的xml scheme各种命名空间以及相应的定义文件地址：<http://www.springframework.org/schema>

导入util和p命名空间，xsi:schemaLocation指定了用于解析和校验xml的定义文件位置。

```

1 <beans xmlns="http://www.springframework.org/schema/beans"
2   xmlns:util="http://www.springframework.org/schema/util"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xmlns:p="http://www.springframework.org/schema/p"
5   xsi:schemaLocation="http://www.springframework.org/schema/beans
6   http://www.springframework.org/schema/beans/spring-beans.xsd
7   http://www.springframework.org/schema/util
8   http://www.springframework.org/schema/util/spring-util-4.3.xsd">

```

顶 跟

1 2

- 上一篇 终端会话管理工具tmux
- 下一篇 Spring全家桶(二)Bean之间的关系、自动装配、作用域和使用外部文件

相关文章推荐

- | | |
|---|--|
| <ul style="list-style-type: none"> • spring全家桶+redis+nginx+activeq+solr商城项目 • MySQL在微信支付下的高可用运营--莫晓东 • 【React全家桶入门之五】组件化表单/表单控件 • 容器技术在58同城的实践--姚远 • 【React全家桶入门之六】渲染用户列表 • SDCC 2017之容器技术实战线上峰会 • 【React全家桶入门之十三】Redux中间件与异步action • SDCC 2017之数据库技术实战线上峰会 | <ul style="list-style-type: none"> • Spring全家桶(四)Bean的生命周期 • 腾讯云容器服务架构实现介绍--董晓杰 • Spring全家桶(八)AOP核心思想与AspectJ 5种类型... • 微博热点事件背后的数据库运维心得--张冬洪 • 【React全家桶入门之七】提取布局组件 • Spring全家桶(五)Bean的多种配置方法 • Spring全家桶(六)必知必会的java注解技术 • 【React全家桶入门之三】基本的用户添加 |
|---|--|

[查看评论](#)

暂无评论

您还没有登录,请[登录](#)或[注册](#)

* 以上用户言论只代表其个人观点, 不代表CSDN网站的观点或立场

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

[网站客服](#) [杂志客服](#) [微博客服](#) webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved

